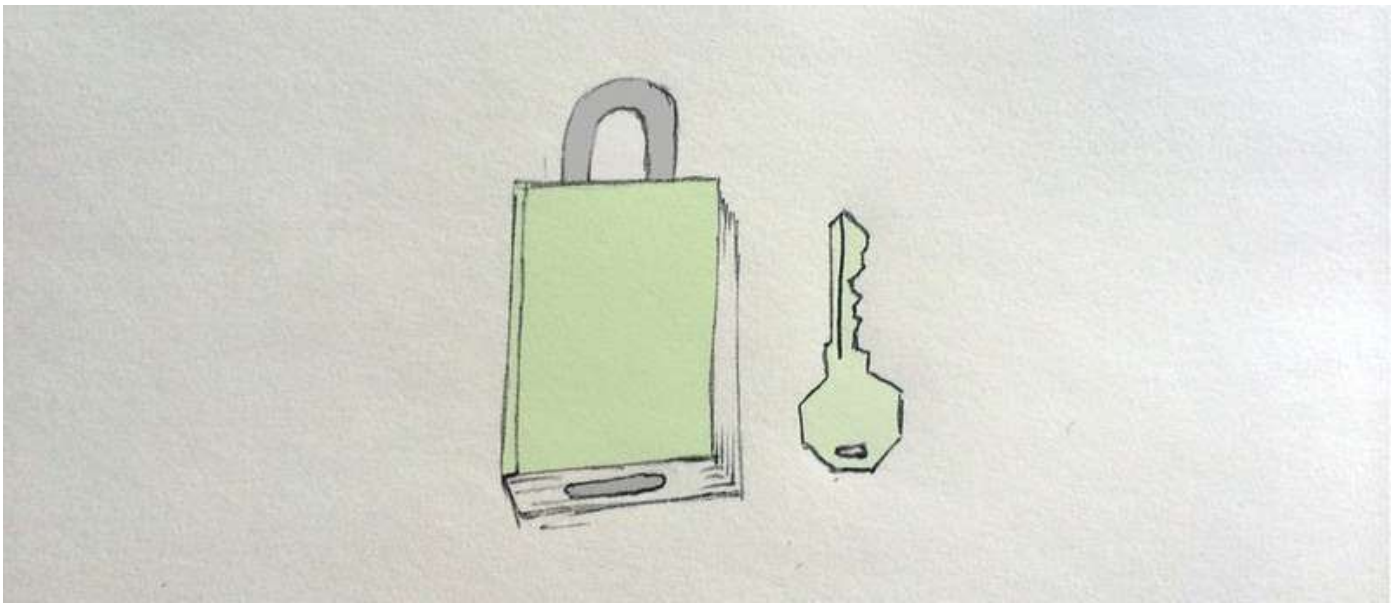
ARTÍCULOS DE TECNOLOGÍA

Tipos de Autenticación: Contraseña, Token, JWT, Dos Factores y Más.

A

Andre Bessa

5 dias atras



Antes de empezar a hablar sobre los tipos de autenticación, es importante preguntarnos: ¿por qué es tan necesaria esta técnica en aplicaciones de cualquier tipo?

Imaginemos el siguiente escenario: es 23 de diciembre y por la la fiebre de fin de año, para no enfrentar la aglomeración de personas en una *mega tienda*, eliges comprar en la tienda de *comercio electrónico* de tu opción. En muchos casos, será necesario que confirmes en el sitio web de la tienda que eres tú mismo, muy probablemente ingresando tu nombre de usuario y contraseña. La razón de esto es bastante simple: la seguridad.

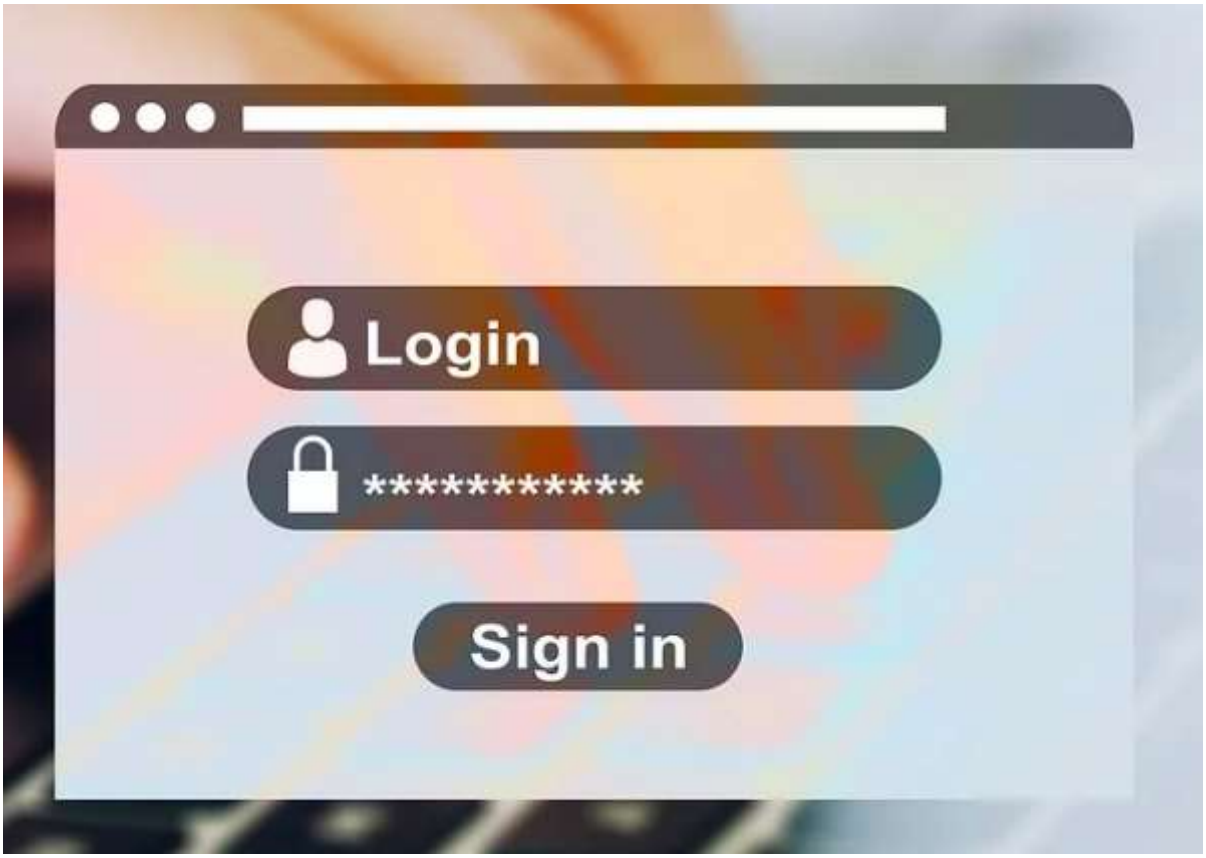
Todo el mundo quiere sentirse seguro al utilizar un sistema, especialmente cuando se tienen que proporcionar datos confidenciales como la identidad o información bancaria. A través de los métodos de autenticación, se puede prevenir el fraude y el mal uso de la información en entornos digitales.

Por lo tanto, analicemos algunos tipos de autenticación que se pueden implementar en nuestros sistemas.

Tipos de Autenticación

Recuerda que, dependiendo de la complejidad y los requisitos de seguridad de tu aplicación, puedes optar por utilizar más de un método de autenticación. Los tipos más utilizados durante el desarrollo son los siguientes:

Autenticación por usuario y contraseña



El uso de PINs, nombre de usuario o contraseña, es una de las formas más básicas y sencillas de implementar la autenticación en los sistemas, asumiendo que solo el usuario dispone de esta información de acceso.

En este modelo, la protección de los datos depende del usuario, ya que cada uno elige qué contraseña o PIN tendrá. Sin embargo, un problema es que los hackers pueden capturar esta información, lo que vulnera la seguridad del esquema de seguridad propuesto.

Autenticación por Biometría

Este tipo de autenticación se basa en la lectura de alguna característica física única del individuo, como una huella dactilar, un escaneo del iris o incluso la voz. Esta es una forma muy efectiva para que los sistemas validen que la persona que solicita acceso es quien dice ser.

Autenticación de dos factores

En este método, la idea principal para la autenticación es agregar una capa más de seguridad para acceder a nuestras aplicaciones. Por lo tanto, el usuario necesita información adicional para terminar de acceder.

Por ejemplo, en un primer paso la aplicación solicita un pin. Además, reenvía un código que complementará la autenticación por algún canal de comunicación del usuario, como el número celular o el correo electrónico.

Un punto de atención en este enfoque es que el usuario necesariamente debe tener disponible el canal adicional en el proceso de inicio de sesión. Si tenemos que ingresar el nombre de usuario y la contraseña en los campos de una página web en la primera etapa de acceso y luego esperar el envío de un código por SMS al celular, necesitamos tener el dispositivo con nosotros para completar el acceso.

Autenticación por sesión

Este fue uno de los primeros métodos de autenticación, creado en el desarrollo temprano de las aplicaciones web. Ampliamente utilizado hasta el día hoy, en este modelo el usuario puede autenticarse con usuario y contraseña o por algún otro método.

El servidor, a su vez, crea una sesión en su memoria o base de datos y devuelve la información del usuario a través de una cookie con el identificador de la sesión creada.

En la siguiente solicitud, se pasa el identificador de sesión y el servidor da acceso al recurso solicitado o realiza algún otro tipo de manipulación con respecto a la cuenta autenticada.

Autenticación por Token

En este método luego de tener un usuario y contraseña validados por el servidor, se creará un token que el usuario recibirá como respuesta y que le permitirá acceder a algún recurso. El estándar adoptado por una gran cantidad de aplicaciones web en la actualidad es el formato JWT (JSON Web Token) y hará que el token se firme correctamente para autenticar la solicitud a un recurso en el servidor.

Es importante tener en cuenta que la información del usuario no se guardará en el servidor. Están escritos en el token, que generalmente tiene un período de vencimiento corto, alrededor de 10 minutos. El tiempo dependerá de los requisitos de seguridad de la aplicación.

Si alguna vez caduca un token, una solución sería pedirle al servidor un nuevo token válido. Esto implicaría un nuevo proceso de autenticación de inicio de sesión y contraseña u otro método elegido. Sin embargo, no es práctico que la aplicación solicite nuevamente el nombre de usuario y la contraseña en cada fecha de vencimiento. Una estrategia para lidiar con esta situación sería usar un token de actualización.

Como solución común, se acostumbra enviar tanto el token de acceso como el token de actualización, tan pronto como se realiza la autenticación en la aplicación. Sin embargo, el token de actualización tiene un tiempo de caducidad más largo y, en este caso, no almacenamos ninguna información del usuario.

tra característica del token de actualización es que solo se puede usar, por lo que la siguiente solicitud envía un token y un nuevo `token de actualización`.

Autenticando por sesión VS Token JWT

Habiendo presentado las características de estos dos métodos, ahora veremos una comparación entre ellos.

Durante la autenticación de la sesión, el servidor mantiene el estado de la sesión con la información del usuario, que se puede almacenar en una base de datos o en la memoria (`Stateful`). Con esta estrategia, podemos encontrarnos con problemas como límites de hardware. El consumo excesivo de memoria puede incluso hacer que la máquina se bloquee, dada la cantidad de llamadas al `Garbage Collector`,(recolector de elementos no utilizados).

Cuando usamos la autenticación por token *JWT*, el escenario cambia un poco, ya que la información ya no se almacena en el servidor (`Stateless`), sino en el token, en la máquina del cliente. El token llevará toda la información necesaria para la autenticación, como la identificación del usuario, la firma, la fecha de vencimiento e incluso el método de autenticación utilizado.

Pero de manera práctica, ¿cómo diferenciar los dos modelos de autenticación? En principio son muy similares. Lo que los diferenciará de manera más efectiva será mantener o no el estado en el servidor (`stateless/stateful`) lo que afectará la forma en que implementamos la escalabilidad, para que nuestro sistema pueda manejar una mayor carga de trabajo.

Escalando una aplicación con seguridad

En el ciclo de vida de una aplicación de cualquier tipo, es muy común que comience a demandar más recursos para su funcionamiento. Este crecimiento debe ser ordenado y respaldado de manera que no afecte la utilización del sistema, lo que llamamos **escalabilidad**.

La implementación de la seguridad a través de uno de los métodos de autenticación presenta algunos desafíos para la escalabilidad de la aplicación. Si optamos por utilizar sesiones, el control tiende a ser complejo, ya que el estado puede estar repartido entre varios servidores y/o instancias.

En este contexto, es común tener una figura de balanceador de carga, responsable de distribuir la carga de solicitudes a los recursos entre los demás servidores que mantienen la aplicación.

En el caso de la autenticación por sesión, ¿cómo podemos garantizar que se devuelva la sesión correcta al cliente? Trabajar con el concepto Sticky Session puede ser útil, ya que en escenarios de equilibrio de carga, esta técnica vincula una sesión de usuario a un servidor/instancia específicos.

Escalar una aplicación es muy importante y debemos utilizar métodos de autenticación que cumplan con este requisito, garantizando así una mayor seguridad en el uso de los sistemas por parte de nuestros usuarios.

Conclusión

Independientemente del método adoptado, lo importante es que al diseñar una solución de software, el desarrollador “esté atento” a la implementación de métodos de autenticación que permitan la escalabilidad de la aplicación con el menor inconveniente, además de un mantenimiento sin estrés.

André Bessa

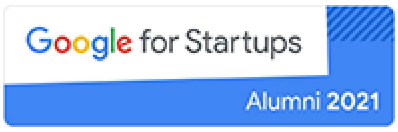
Soy programador e instructor de programación usando C# y .NET. Soy Licenciado en Sistemas de Información y con especialización en Ingeniería de Software y Estudios Superiores. Tengo experiencia en desarrollo usando Java, PHP, PostgreSQL y MySQL, además de trabajar con soporte e implementación de sistemas. Siempre busco aprender más, también me gusta contribuir a la enseñanza y difusión de la tecnología. En las horas de ocio, maratoneo alguna serie, leer historias de héroes.

Traducido para Alura Latam por **Luis Puig**.

ARTÍCULOS DE TECNOLOGÍA

ALIADOS

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science
Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps
Docker | Linux

Cursos de Innovación y Gestión